Assignment 3: Gathering, Scraping, Munging and Cleaning Data

Submission by: Abhishek Nair (NUID: 002724762) and Hrishikesh Sanjay Pawar (NUID: 002707307)

Topic: University Marketplace Database

Topic description:

When Abhishek and I arrived in Boston for our Master's program, it was an arduous task among 500+ students to find used essential items like study tables, chairs, bed frames, study lamps, etc. University Marketplace Database project is a database of used or pre-owned essential items for peer to peer buying/selling created for university students. The project addresses two issues:

1. Mostly buying/selling of used items is done on instant messengers like Whatsapp via multiple groups. This creates a decentralized environment where it is difficult to keep track of all the buying/selling opportunities.
2. On general platforms like Facebook Marketplace or Ebay, the buyers/sellers might not be trustworthy and/or may be located at an inconvenient distance from the university.

We are trying to resolve these issues by creating a centralized database of pre-owned essential items that can be accessed at a single place. The marketplace will be segregated based on universities to make sure that buyers and sellers are located at a reasonable distance. We will be using users' university email address for the verification purpose. Along with the general details of the product, distance of each product from the buyers will be computed based on the lag-long coordinates. This will help buyers decide which seller to buy from. We are also planning to incorporate a premium subscription feature into the project for sellers. This subscription will enable the sellers to list their products at the top of the search results

Data Sources

Source for university dataset: https://public.opendatasoft.com/explore/dataset/us-colleges-and-universities/export/

Source for student dataset:
https://github.com/ShapeLab/ZooidsCompositePhysicalizations/blob/master/Zooid_Vis/bin/data/student-dataset.csv

Source for products, category, and feedback dataset:
https://www.kaggle.com/datasets/PromptCloudHQ/flipkart-products

Source for orders dataset: https://github.com/pawarbi/datasets/blob/master/Orders.csv

SQL to insert the data into database

Category Table:

```
insert into twitter_schema.category(category_id, category_name) values
(1,'Clothing');
insert into twitter_schema.category(category_id, category_name) values
(2,'Furniture');
```

Product Table:

```
insert into twitter_schema.product(product_id, product_name, category_id) values
(1,'Alisha Solid Womens Cycling Shorts',1);
insert into twitter_schema.product(product_id, product_name, category_id) values
(2,'FabHomeDecor Fabric Double Sofa Bed',2);
```

Feedback Table:

```
insert into twitter_schema.feedback(feedback_id, feedback_score,comments) values
(1,1,'Worst Product');
insert into twitter_schema.feedback(feedback_id, feedback_score,comments) values
(2,1,'Worst Product');
```

Student Table:

```
insert into twitter_schema.student(student_id,
university_id,university_name,first_name,last_name) values (70,'Northeastern
University',’MA’,’Boston’);
insert into twitter_schema. student (student_id,university_id,
university_name,first_name,last_name)  values (34,21,Harvard
University,’MA’,’Boston’);
```

Buyer Table:

```
insert into twitter_schema.buyer(buyer_id, student_id) values (99,12);
insert into twitter_schema.buyer(buyer_id, student_id) values (87,55);
```

Seller Table:

```
insert into twitter_schema.seller(seller_id, student_id,premium_flag) values (11,86);
insert into twitter_schema.seller(seller_id, student_id,premium_flag) values (24,52);
```

Use Cases of queries that are particular to our domain:

1. University wise number of students
Description: Seller views count of students in each university
Actor: Seller
Precondition: There should be students enrolled in at least one univeristy
Steps-
Actor action: Seller requests to view the student count for all univeristies
System Responses: Displays the number of students in each university
Post Condition: Seller is successfully able to view the student count for all univeristies
Alternate Path: There are no students enrolled in any university
Error: No students enrolled in universities

SQL -

```
SELECT university_name, COUNT(student_id) student_count
FROM University U
INNER JOIN
Student S
ON S.university_id = U.university_id
GROUP BY 1;
```

2. View premium seller with highest total sales amount
Description: Buyer views premium seller with highest business done (sales)
Actor: Buyer
Precondition: There should be premium sellers with at least one order
Steps-
Actor action: Buyer requests to view the premium seller with highest sales mount
System Responses: Displays the premium seller with highest total sales
Post Condition: Buyer is successfully able to view the premium seller with highest sales mount
Alternate Path: There are no premium sellers with any orders
Error: No premium sellers having at least one order

SQL -

```
SELECT T.seller_id as seller_id, SUM(price) as Total_Sales
FROM Twitter_Order T
INNER JOIN
Seller S
ON S.seller_id = T.seller_id
WHERE premium_flag = 1
GROUP BY seller_id
ORDER BY Total_Sales DESC
LIMIT 1;
```

3. Which State has the most number of buyers?
Description: Seller views state with most number of buyers
Actor: Seller
Precondition: There should be students registered with at least one university
Actor action: Seller requests to view the state with most number of buyers
System Responses: Displays the state with highest buyers
Post Condition: Seller is successfully able to view the state with most number of buyers
Alternate Path: There are no students enrolled in any university
Error: No universities or states having any enrolled students

SQL -

```
SELECT U.state as State, COUNT(B.buyer_id) Buyer_Count
FROM University U
LEFT JOIN
Student S
ON S.university_id = U.university_id
LEFT JOIN
Buyer B
ON B.student_id = S.student_id
GROUP BY State
ORDER BY Buyer_Count DESC
LIMIT 1;
```

4. What is the average number of transactions per seller for sellers without premium subscription?
Description: Buyer views average number of transactions for each seller without premium subscription
Actor: Buyer
Precondition: At least one seller should have a buy or sell transaction
Actor action: Buyer requests to view the average number of transaction per seller without premium subscription
System Responses: Displays the average number of transactions per seller without premium subscription
Post Condition: Buyer is successfully able to view the average number of transactions each seller without premium subscription
Alternate Path: There are no seller having any transactions
Error: No seller has performed any transaction

SQL –

```
SELECT AVG(Order_Count) Avg_Transactions_Per_Seller
FROM
(SELECT T.seller_id as seller_id, COUNT(T_order_id) Order_Count
FROM Twitter_Order T
INNER JOIN
Seller S
ON S.seller_id = T.seller_id
WHERE S.premium_flag = 0
GROUP BY 1) D;
```

5. View the seller with highest number of 5-star feedbacks
Description: Buyer views seller having highest number of 5-star ratings or feedbacks
Precondition: There must be at least one order with some feedback
Actor action: Buyer requests to view the seller with highest rating
System Responses: Displays the Seller with highest sales feedback ratings
Post Condition: Buyer is successfully able to view the seller with highest rating
Alternate Path: There are no orders having a feedback rating
Error: No orders having feedback rating present in the system

SQL -

```
SELECT seller_id, COUNT(T_order_id) 5_STAR_TXN_COUNT
FROM Twitter_Order T
LEFT JOIN
Feedback F
ON F.feedback_id = T.feedback_id
WHERE feedback_score = 5
GROUP BY 1
ORDER BY 5_STAR_TXN_COUNT DESC
LIMIT 1;
```

6. Use Case 6: Top 10 orders which have received the negative feedback
Description: Seller tweets on the Twitter platform to sell a used/new item.
Actor: Seller
Precondition: When a seller wants to sell an item on the Twitter platform, he needs to sign up first. Also, since our database is catered to university students, we will verify if the user (buyer or seller) belongs to a university by valid their email address in the registration
Steps-
Actor action: Seller creates an account on Twitter
System Responses: If the seller is successfully registered, he/she should be able to tweet and mention which items he needs to sell.
Post Condition: Seller successfully able to tweet on Twitter

SQL –

```
SELECT t.t_order_id, f.feedback_score
FROM twitter_schema.Twitter_Order t
INNER JOIN twitter_schema.Feedback f
ON f.feedback_id = t.feedback_id
ORDER BY feedback_score ASC
LIMIT 10;
```

7. View Category Name and Product Name for a specific product ID

Description: The buyer should be able to view all the products that he/she can buy on the platform.
Actor: Buyer
Precondition: When a buyer wants to buy an item on the Twitter platform, he needs to be able to
successfully sign up first.
Steps-
Actor action: Buyer can view the item that he can purchase on Twitter.
System Responses: If the buyer is meeting all the preconditions, he should be able to view the items
that can be purchased.
Post Condition: Buyer successfully able to view tweets on Twitter.
Alternate Path: The buyer is not able to view products since the system might be down or the product is temporarily unavailable.
Error: If the buyer is not able to view the products, then throw the “Items cannot be viewed during
this time" error.

SQL –

```
SELECT p.product_name, c.category_name
FROM twitter_schema.Product p
INNER JOIN twitter_schema.Category c
ON p.category_id = c.category_id;
```

8. View tags mentioned by the particular twitter_user in a tweet_text
Description: For a particular university, we can see which seller sold the most items.
Actor: User (buyer or another seller)
Precondition: The user needs to sign up first in order to view the seller from the university that sold
most items.
Steps-
Actor action: A user from a particular university can view the user that sold the most items
System Responses: If the User is successfully registered, he/she should be able to view the seller
from the university that sold most items.
Post Condition: The user is successfully able to view the seller who sold most items.
Alternate Path: The user is not able to view sellers which sold most items if the system is down and
an error is thrown.
Error: If the user is not able to view the item, then throw the "Items cannot be viewed during this.

SQL –

```
SELECT t.twitter_handle, t.tweet_text, t.tweet_date, tt.tag
FROM twitter_schema.Tweet t
INNER JOIN twitter_schema.Tweet_tags tt
ON t.tweet_id = tt.tweet_id;
```

9. Who is the source and the target user mentioned by the particular twitter user in a tweet

Description: The user should be able to view a product below a certain price
Actor: User (Buyer or seller)
Precondition: The user or a buyer needs to sign up first in order to view the product below a particular price.
Steps-
Actor action: A user or buyer from a particular university can view a product below a particular
price.
System Responses: If the User or buyer is registered and the system is not down, he/she should be
able to view the products below a particular price.
Post Condition: User or buyer successfully able to view the products below a particular price.
Alternate Path: The user or buyer is not able to view the products below a particular price and an
error is thrown.
Error: If the user or buyer is not able to view the products below a particular price then throw the
“Items cannot be viewed during this time" error.

SQL -

```
SELECT t.twitter_handle, t.tweet_text, t.tweet_date, tm.source_user, tm.target_user
FROM twitter_schema.Tweet t
INNER JOIN twitter_schema.Tweet_Mentions tm
ON t.tweet_id = tm.tweet_id;
```

10. Who are the sellers that are students and have they enrolled for a premium option?
Description: The user should be able to view the top 3 order id which has received the worst feedback.
Actor: User (Buyer or seller)
Precondition: The user needs to sign up first in order to view the top 3 order IDs that received the
worst feedback.
Steps-
Actor action: The user selects the top 3 order id that has received the worst feedback.
System Responses: If the system is not down, he/she should be able to view the top 3 order id which
has received the worst feedback.
Post Condition: User was successfully able to view the top 3 order IDs which have received the
worst feedback.
Alternate Path: The user is not able to view the top 3 order id that has received the worst feedback
and an error is thrown.
Error: If the user is not able to view the top 3 order id which has received the worst feedback, then
throw "Items cannot be viewed during this time" error.

SQL –

```
SELECT student.first_name, student.last_name, seller.premium_flag
FROM twitter_schema.Student student
INNER JOIN twitter_schema.Seller seller
ON student.student_id = seller.student_id
where seller.premium_flag = 1;
```